OIPE JC11
APR 0 2 2004
PATENT & TRADEMARK OFFICE

PATENT
Customer No. 22,852
Attorney Docket No. 08516.0007

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of:	
Thomas AREND	Group Art Unit: 2184
Application No.: 10/697,434	Examiner: Unknown
Filed: October 31, 2003	
For: IDENTIFYING SOLUTIONS TO COMPUTER PROBLEMS IN MAIN SYSTEM BY SERVICE SYSTEM IN DISTRIBUTED SYSTEM LANDSCAPE	

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

CLAIM FOR PRIORITY

Sir:

Under the provisions of Section 119 of 35 U.S.C., Applicant hereby claims the benefit of the filing date of European Patent Application Number 02024533.8, filed October 31, 2002, for the above identified United States Patent Application.

In support of Applicant's claim for priority, a certified copy of the priority application is filed herewith.

Respectfully submitted,

FINNEGAN, HENDERSON, FARABOW,
GARRETT & DUNNER, L.L.P.

Dated: April 2, 2004

By: ____________________
C. Gregory Gramenopoulos
Reg. No. 36,532

2410

2002P10028EP

Europäisches Patentamt | **European Patent Office** | **Office européen des brevets**

Bescheinigung | Certificate | Attestation

Die angehefteten Unterlagen stimmen mit der ursprünglich eingereichten Fassung der auf dem nächsten Blatt bezeichneten europäischen Patentanmeldung überein.

The attached documents are exact copies of the European patent application described on the following page, as originally filed.

Les documents fixés à cette attestation sont conformes à la version initialement déposée de la demande de brevet européen spécifiée à la page suivante.

Patentanmeldung Nr. | **Patent application No.** | **Demande de brevet n°**

02024533.8

Der Präsident des Europäischen Patentamts;
Im Auftrag

For the President of the European Patent Office

Le Président de l'Office européen des brevets
p.o.

R C van Dijk

DEN HAAG, DEN
THE HAGUE, 11/12/02
LA HAYE, LE

EPA/EPO/OEB Form 1014 - 02.91

Europäisches Patentamt | European Patent Office | Office européen des brevets

Blatt 2 der Bescheinigung
Sheet 2 of the certificate
Page 2 de l'attestation

Anmeldung Nr.:
Application no.:
Demande n°:
02024533.8

Anmeldetag:
Date of filing:
Date de dépôt:
31/10/02

Anmelder:
Applicant(s):
Demandeur(s):
SAP Aktiengesellschaft
69190 Walldorf
GERMANY

Bezeichnung der Erfindung:
Title of the invention:
Titre de l'invention:
Identifying solutions to computer problems in main system by service system in distributed system landscape

In Anspruch genommene Prioriät(en) / Priority(ies) claimed / Priorité(s) revendiquée(s)

Staat: State: Pays:	Tag: Date: Date:	Aktenzeichen: File no. Numéro de dépôt:

Internationale Patentklassifikation:
International Patent classification:
Classification internationale des brevets:
/

Am Anmeldetag benannte Vertragstaaten:
Contracting states designated at date of filing:
Etats contractants désignés lors du depôt:
AT/BE/CH/CY/DE/DK/ES/FI/FR/GB/GR/IE/IT/LI/LU/MC/NL/PT/SE/TR

Bemerkungen:
Remarks:
Remarques:

EPO - Munich
55
31. Okt. 2002

IDENTIFYING SOLUTIONS TO COMPUTER PROBLEMS IN MAIN SYSTEM BY SERVICE SYSTEM IN DISTRIBUTED SYSTEM LANDSCAPE

01 <u>Field of Invention</u>

02 The present invention generally relates to data processing by computer systems, programs, and methods. More particularly, the invention relates to evaluating and solving problems. The computer systems are distinguished into main, auxiliary and service systems.

03 <u>Background</u>

04 Electronic data processing uses integrated and distributed computer systems with complex architecture. Coupling different computers over networks (e.g., Internet) enhances functionality but adds complexity and increases maintenance.

05 Each computer system operates in the complexity of hardware (e.g., computers and network) and software (e.g., operating systems, applications, databases).

06 Problems are deviations from the predefined operation of the computer system that are caused by malfunction of hardware or software or by improper input by the user. To name a few examples, components like processors suddenly fail, applications occasionally provide wrong results, and users sometimes manipulate data.

07 Problems often remain hidden from the user. Once detected, the user engages in problem solving. For example, the user reads documentation papers, activates help functions (e.g., predefined advices, often obtained via online services), looks up in databases to identify advices ("notes"), makes

experiments, or tells problem symptoms to specialists (e.g., through phone hotline, email, Internet portal).

08 A majority of users relies on passive assistance; only a minority actively solves the problem. There are further challenges: For example, sensitive data remains with the authorized user but is shielded from specialists (data protection); users and specialists might introduce further errors. In any case, problem solving remains time consuming and expensive.

09 Further, heterogeneous system landscapes have systems that differ for example, by manufacturer, release version, and application. Each difference increases the number of potential problems and corresponding solutions. Selecting solutions becomes critical.

010 There is a need to improve problem solving by mitigating disadvantages of the prior art.

011 Brief Description of the Drawings

012 FIG. 1 illustrates a simplified block diagram of a computer system with a main system and an auxiliary system according to the present invention;

013 FIG. 2 illustrates the system of FIG. 1 with more detail;

014 FIG. 3 illustrates a service module in the auxiliary system with more detail;

015 FIG. 4 illustrates an acquisition module in the auxiliary system with more detail;

016 FIG. 5 illustrates a knowledge module in the auxiliary system with more detail;

017 FIG. 6 illustrates an inference module in the auxiliary system with more detail;

018 FIG. 7 illustrates a first distributed system landscape with the main and auxiliary systems coupled to a service system;

019 FIG. 8 illustrates a second distributed system landscape with 2 main systems coupled to a service system;

020 FIG. 9 illustrates a simplified flowchart diagram of a method for operating the main, auxiliary and service systems;

021 FIG. 10 illustrates a simplified scenario that considers interaction, and thereby distinguishes automatically problem evaluating and semi-automatically problem evaluating;

022 FIG. 11 illustrates a simplified scenario that considers primary and secondary context;

023 FIG. 12 illustrates a simplified scenario that considers the distribution of problem collecting and solution processing in the system landscapes;

024 FIG. 13 illustrates further simplified scenarios;

025 FIG. 14 summarizes various aspects of the present invention by concentrating on an inference module; and

026 FIG. 15 illustrates a simplified block diagram of a computer system in general for that the present invention can be implemented.

027 <u>Detailed Description</u>

028 An exemplary implementation for the invention uses the well-known system R/3. R/3 is commercially available from SAP Aktiengesellschaft Walldorf (Baden, Germany, "SAP"). Organizations (i.e. SAP customers) use enterprise resource planning applications ("ERP applications", or "business applications") to organize information in a variety of fields, such as supply chain management (SCM), customer relationship

management (CRM), financials, human resources (HR), enterprise portals, exchanges, technology, product lifecycle management (PLM), supplier relationship management (SRM), business intelligence, business intelligence, mobile business, hosted solutions, small and midsize business, and industry solutions.

029 ABAB/4 is the well-known programming language used by SAP to define transactions for applications. R/3 and ABAB/4 is documented by a variety of reference books, such as:

030 Gareth M. de Bruyn, Robert Lyfareff, Ken Kroes: "Advanced ABAP Programming for SAP". Prima Publishing 1999. ISBN 0-7615-1798-7.

031 Bernd Matzke: "Programming the SAP R/3 System". Addison-Wesley, 1997. ISBN 0-201-92471-4.

032 Jonathan Blain, ASAP World Consultancy: "Special Edition Using SAP R/3, Third Edition. Que. 1999. ISBN 0-7897-1821-9.

033 A detailed description of a computer system in general and a list of reference numbers are provided as the end of the specification.

034 In short, according to the invention, a computer system has a main system to execute an application (A) in cooperation with a human user and has an auxiliary expert system to evaluate problems (P) in the main system. Optionally, the problems are solved by predefined instructions.

035 Auxiliary systems are distributed in a landscape of physically different computer systems (i.e. heterogeneous landscape) that exchange knowledge representations (R) and solutions (S). The computer systems communicate by a network. Due to the exchange of representations and solutions (instead of data), problems on a local system (i.e. main system) are evaluated by a remote

system (i.e. auxiliary system, service system). The remote system returns a set of up-to-date knowledge representations and thereby enables the local system to evaluate the problem locally. The remote system can also evaluate the problems and return solutions to the local system.

036 The present invention enables the user to actively solve problems in the main system mostly without asking for advice by human specialists. Applying the invention saves time and quickly returns the main system back to normal operation. Applying further features effectively escalates problem evaluation to the service system. Involving human specialists (often expensive) is only required as a last remedy.

037 FIG. 1 illustrates a simplified block diagram of a computer system with main system 200 and auxiliary system 300 according to the present invention.

038 Computer system 200/300 has main system 200 to execute application A in cooperation with human user 1000. Auxiliary system 300 evaluates problems P in main system 200. Auxiliary system 300 has service module 310 to collect problem related data D from main system 200, acquisition module 320 to acquire knowledge representations R, knowledge module 330 to store knowledge representations R, inference module 340 for processing problem related data D with knowledge representations R to identify solutions S and for forwarding the solutions S through service module 310 to main system 200. Although the figures explain the functionality by one main and one auxiliary system, the systems are distributed to a plurality of physically different systems in a landscape.

039 Auxiliary system 300 finds problem related data D by evaluating the problem environment in main system 200: date, time, memory

usage, data objects, software modules of application and operating system and the like.

040 FIG. 2 illustrates main system 200 and auxiliary systems 300 with more detail. Main system 200 has a client/server configuration with database 210 (preferably, relational database), application server 220 and front-end server 230.

041 The following refers to an exemplary implementation: Main system 200 and auxiliary system 300 are implemented by an R/3 type system. System 200 performs an ERP application. The ERP application is defined by instructions that have common keywords, common syntax and common semantic with environments selected from the group of: ABAB/4, Java 2 Platform Enterprise Edition (J2EE), and.NET framework. Auxiliary system 300 uses the client/server configuration (210, 220, 230) of main system 200: the modules of auxiliary system 300 are distributed such that service module 310, acquisition module 320, knowledge module 330, inference module 340 are arranged in parallel to application server 220 and to database 210. Front-end server 230 operates as user-interface both for main system 200 and auxiliary system 300. In other words, database 210 implements a storing function, application server 220 implements the application (A, cf. FIG. 1) and front-end server 230 implements presentations (e.g., user interface). The module distributions can be modified. For example, knowledge module 330 can be part of database 210. Internet communication is used between application server 220 and front-end server 230. Internet communication is implemented by well-known techniques (e.g., TCP/IP, HTML, and HTTP).

042 Having introduced main system 200 and auxiliary system 300 in general, the following sections describe the modules with more

detail.

043 FIG. 3 illustrates service module 310, especially its cooperation with main system 200 (dashed frame) in the exemplary implementation. Service module 310 makes basis service functions of main system 200 available for auxiliary system 300 (cf. FIG. 2). Basis service functions are: ABAP/4 workbench, administration, authorizations, batch input, data dictionary, dialog control, framework, graphical user interface, application program interface, and job. Basis services are explained in the above-cited reference books.

044 Service module 310 cooperates with main system 200 to obtain problem related data D for auxiliary system 300. Service module 310 cooperates with database 210 to test the existence of objects: The problem related data D comprises information about existence and non-existence of the objects. The objects are related to application A (in server 220).

045 Service module 310 cooperates with database 210 to obtain the content of a table entry as problem related data D. Service module 310 records events in the operating system of main system 200 by writing to database 210. Service module 310 records problem related data D obtained from data consistency check operations of main system 200 (e.g., application server 220). Consistency checks determine consistency (non-consistency) of data in database 210, especially in the database tables: Entries in the table-body should be consistent with the entries in the table-header. For example, the body holds zip-code numbers below headers "zip-code". Service module 310 instructs front-end server 230 to provide dialogs with user 1000. Service module 310 provides remote function call (RFC) connections with further auxiliary systems (with similar modules), such as with a service system (cf. FIG. 6).

046 Service module 310 monitors application server 220 and database 210 according to instructions from inference module 340.

047 FIG. 4 illustrates acquisition module 320 for the exemplary implementation. Acquisition module 320 also modifies knowledge representations R. Acquisition module 320 interacts with knowledge engineer 1001, as illustrated, through tree 322 on a graphical user interface. Acquisition module 320 uses tree 322 to represent the knowledge representations R as a semantic net. Engineer 1001 may use well-known edit or drag and drop techniques to manipulate the representations.

048 Knowledge engineers are, for example, (a) software developers who are familiar with main system 200, (b) technical writers who write documentations for customers, and (c) persons that have gathered experience as being a user (cf. 1000 in FIG. 1). Tree 322 assists engineer 1001 to modify knowledge representations. As in the example, tree 322 represents a rule relating to a patch type. The rule has a query step (Download OK?) and conditional steps. A patch is distinguished by its source between Compact Disk, OSS (online service system, cf. reference books), and Internet. Depending on the source, different advices can be defined. Also, user 1001 is invited to modify tree 322 by inserting icons from an icon tray ("insert objects").

049 FIG. 5 illustrates knowledge module 330 for the exemplary implementation. Knowledge module 330 stores knowledge representations R by classifying into context, for example, business transactions as part of the application A, executable programs as part of the application A, and hierarchy level within the application A.

050 Optionally, knowledge module 330 uses lexicon 331 to distinguish versions of main system 200 (e.g., versions "1.0" and "2.0"). Lexicon 331 defines parameters Pa (for a particular version) and knowledge representations R (for all versions). The figure illustrates the different parameters by different hatching. Distinguishing is very convenient if auxiliary system 300 serves 2 or more main systems 200 that have different version (i.e. release or language). For example, main systems 200 might differ in their table definitions due to different release dates. Distinguishing through parameters for equal representations helps to keep the number of knowledge representations at a convenient level. Useful is also to distinguish between different natural languages: For example, while a first main system communicates with its users in German; a second main system communicates with its users in English; both mains systems are supported by a common auxiliary system that provides dialog texts in English or German. Knowledge module 330 makes the knowledge representations R selectively available or non-available according to a selected context or version.

051 Knowledge module 330 distinguishes context with primary context and secondary context, wherein the secondary context is referenced from the first context. The first context can refer to the second context, the second context can refer to the first context, or both contexts can refer to each other. Knowledge module 330 selects knowledge representations R to be considered by inference module 340 according to the context of a current transaction by the application server 220. Selected context is selected by user 1000 or by a predefined rule. For example, the context is selected from: system and program performance, background processing, OCS and patches, data

dictionary, printer problems, remote function calls and connectivity, R/3 reporting, and security and administration.

052 For example, the first context is defined by the application with a transaction "Patch Manager". Problem P and data D are "Patch not found". Knowledge representations R have hints to find a storage location for patches (e.g., a directory or a server). But processing does not result in a solution S. The problem remains unsolved. However, the link "Transport" refers from the first context to the second context. The second context leads to knowledge representations R to software installing procedures. Processing with these representations R leads to solutions S.

053 Knowledge module 330 is adapted to receive regular updates of the knowledge representations R (arrow symbol). The updates can originate, for example, from a service system (cf. FIG. 5) that acts as further auxiliary system. Knowledge module 330 stores knowledge representations R in database 210 with entries for specific problem P symptoms and corresponding solutions S.

054 Knowledge module 330 stores knowledge representations R that point to predefined solution identification rules in database 210 (or in module 330 itself). The solution identification rules are provided in a meta language. The meta-language is derived from ABAP/4.

055 The rules usually identify action, object, arguments and result location. An example for meta language is, for example, a 2-line rule for storing data. The first line reads as "CALL" (action), "FUNCTION" (object), "GET_FILE_PATH" (argument), and "FILE_PATH" (result location) to find a file directory (path) and to keep the file directory in a first variable ("FILE PATH"). The second line reads as "CHECK_EXIST" (action), "FILE" (object), "<FILE_PATH>/rfc.trc" (argument) and "EXIST" (result location) to check the existence of file "rfc.trc" in the

directory and to write existence/absence result into a variable ("EXIST").

056 Providing R in meta-language in convenient for automatically processing. Markup languages (e.g., XML) are also useful. It is an advantage that R can also be provided partially or completely in natural languages (e.g., English) for "processing" by a human.

057 Knowledge module 330 generates a structured set of problem solving strategies, such as so-called "troubleshooting guides". These are strategies for consideration by a specialist or by the user.

058 Knowledge module 330 generates solution identification rules with computer instructions that the computer utilizes to automatically solve the problem. Preferably, knowledge module 330 distinguishes error classes (details below).

059 Sets of semantically related solution identification rules are grouped together, such as rules to find the problem "inconsistency in a table", and rules to find the corresponding code to "automatically re-arrange the table" (i.e. utilizing the solution). Tables in the database 210 are not only used to organize data for application A, but also convenient for knowledge module 330 to stores knowledge representations R. In that case, the tables are provided prior to activating auxiliary system 300.

060 The following is an example for using knowledge representations R, context and check lexicon: The knowledge representations form a set of R1, R2, R3, ..., R16, ..., R99 (consecutively numbered). Each R is defined by meta-language, such as "CHECK PRINTER CONNECTION" for R16.

061 Context are subsets of representations for that relate to problem classes, such as context PRINTER with R5, R6 and R16.

The check lexicon lists details for knowledge representations depending on versions of main system 200 (optionally versions of application A). R16 for version 3.0 testing an IP connection by a standard "PING PRT" command to the printer; for version 3.1 calling a dedicated test function (in auxiliary system 300); for version 4.0 instructing the printer to print a test page.

062 When processing problem data D such as "printing not possible" for main system 300 of version 3.0, auxiliary system 200 extract the context set PRINTER from all R, checks the lexicon for applicable details and applies each R in combination with the details (e.g., R16 PING PRT). If a solution is still missing, the context changes, for example, to GENERAL with R1 "CHECK POWER SUPPLY". Applying R1 - this time without distinguishing versions - leads to a success. The printer was not connected to the mains power. The solution S is identified as a message to the user that is forwarded to the user (in the further context of the English language, through the front-end server): "Please connect your printer to the 230 volts power supply."

063 FIG. 6 illustrates inference module 340 in auxiliary system 300 with more detail. Inference module 340 identifies the solutions S from sets of predefined advices (preferably, advices of application A) by a solution identifier.

064 Inference module 340 identifies the solutions by applying the knowledge representations R (to data D) in a predefined order that is, for example, a sequential order (e.g., R1, R2, R3), a hierarchical order (e.g., R1, R21, R21, R31, R32), a dynamically adaptive order in that the order might chance by conditional jumps or the like (e.g., IF THEN).

065 Inference module 340 communicates questions to user 1000 (cf. Q1, Q2, Q3 reservoir). Preferably, the questions are standard questions. Inference module 340 consecutively numbers the questions. Inference module 340 composes the questions from predefined passages that are provided by application server 220. Inference module 340 analyzes the responses that user 1000 enters in natural language (cf. language analyzer).

066 So far the exemplary implementation has been described with main system 200 and auxiliary system 300 that communicate by basis functions and that are implemented by a single R/3 system. Distributing modules of auxiliary system 300 to a client/server system configuration is convenient. The invention is however not limited to that. Further implementations benefit from the following:

(a) Main and auxiliary systems can be distributed to different computer systems (e.g., different R/3 systems; cf. FIG. 7).

(b) A further auxiliary system (here called service system) can provide enhanced problem evaluation capacities (cf. FIG. 7).

(c) One auxiliary system can serve 2 or more main systems (cf. FIG. 8).

d) Applicable knowledge representations can be selected for a particular version of the main system, for example, by maintaining a check lexicon.

e) A first auxiliary system starts to evaluate the problem by first knowledge representations and forwards evaluation results to a second auxiliary system. The second auxiliary system then returns a second (enhanced) knowledge representation to enable the first auxiliary system to finish the evaluation.

067 For convenience of explanation, the following uses the term "system 200/300" collectively for the combination of main system 200 with auxiliary system 300 for main system 200 alone. In other words, auxiliary system 300 is not longer required in any case.

068 FIG. 7 illustrates a first distributed system landscape with system 200/300 coupled to a service system 500 via a network. Service system 500 has auxiliary components with functions that are substantially similar to that of auxiliary system 300: service module 510, acquisition module 520, knowledge module 530, inference module 540 as well as modules for front-end communication. The foregoing description is applicable for these modules as well.

069 Preferably, system 500 operates independently from any main system and does not execute an ERP application. Optionally but not mandatory, service system 500 has a client/server configuration, such as system 500 in an exemplary implementation being an R/3 type system. In comparison to system 200, service system 500 uses knowledge representations R that are enhanced in terms of volume, actuality, and complexity. If required, system 500 also solves problems in auxiliary system 300. In short, system 500 has at least the above-mentioned functions of auxiliary system 300 and serves as the expert system for system 200/300.

070 Service system 500 is conveniently installed at a manufacturer's site (of system 200/300) and communicates with system 200/300 by receiving problem data (D) from system 200/300 and sending control instructions (C) to system 200/300.

071 Service system 500 is - optionally - operated by service engineer 1002 who helps to solve problems in system 200/300. Dynamic enhancement is possible: control instructions C are

conveniently also used to regularly update knowledge module 330 with actual knowledge representations (R, cf. FIG. 5 updates).

072 If system 200/300 is implemented with auxiliary system 300, then auxiliary system 300 acts as a first expert system and service system 500 act as a second expert system. Depending on the severity of the problem (in system 200), problems are evaluated as follows:

073 (1) Auxiliary system 200 solves the problem (i.e. solutions S are identified by system 300).

074 (2) Auxiliary system 200 does not solve the problem but forwards a package with problem P data in combination with preliminary solutions S (i.e., P/S data, based on knowledge representations (R) in system 200) to service system 500. Service system 500 then solves the problem.

075 (3) Auxiliary system 200 does not solve the problem but forwards the P/S data package to service system 500. System 500 uses the P/S data to return further knowledge representations. This enables system 200 to evaluate and solve the problem.

076 (4) Service system 500 does not solve the problem automatically and needs to interact with service engineer 1002 (further analysis by a human technician).

077 FIG. 8 illustrates a second distributed system landscape with main systems 201 and 202 coupled to service system 500. The approach of FIG. 7 has been expanded by adding a further main system. Optionally, service system 500 is operated by a service engineer (cf. 1002).

078 In the exemplary implementation, main system 201 is physically implemented on a first computer; main system 202 (as system 201 also in client/server configuration) is implemented on a second computer, service system 500 is implemented on a third

computer. Auxiliary systems are optionally added to main systems 201 and 202.

079 Main system 201 is adapted to be operated by a first customer (e.g., a first company), service system 500 is implemented by expertise service provider ESP and main system 202 is adapted to be operated by a second customer (e.g., a second company). For example, ESP is the manufacturer of systems 201/500/202 or is a consulting agency. Preferably, main systems 201 and 201 are systems of the same type (e.g., R/3), but have different release versions (i.e. 201 older than 202, or vice versa).

080 Different release versions are distinguished by context groups. (cf. FIG. 5). Such an arrangement is convenient also for main systems that communicate with their users in different natural languages. While problem identification is technically the same in systems 201/202, messages to users (e.g., notes, dialogs) can be in different natural languages.

081 Some or all of the computers are located at physically different locations. Expertise of service system 500 becomes available around the globe. Service system 500 could simultaneously serve main systems 201/202 in different continents around the clock.

082 Having service system 500 physically separated from main systems 201/200 has further beneficial effects: For example, expertise (i.e. knowledge representations R) is shielded from access by main system 201/202; and sensitive data on main systems 201/202 is shielded from access by service system 500.

083 Further distributions of main, auxiliary and service systems are possible. For example, a plurality of main systems can be equipped with auxiliary systems that solve problems for their corresponding main system or forward problem data to service systems.

084 FIG. 9 illustrates a simplified flowchart diagram of method 400 for operating computer system 200/300. Method 400 is also applicable if auxiliary system 300 is replaced by service system 500.

085 As stated above, system 200/300 has main system 200 executing application A in cooperation with human user 1000 and has auxiliary system 300 evaluating problems P in main system 200. According to method 400, auxiliary system 300 performs the following steps: collecting 410 problem related data D from main system 200, acquiring 420 knowledge representations R, storing 430 knowledge representations R, processing 441 problem related data D with knowledge representations R to identify solutions S, and forwarding 442 the solutions S through service module 310 to main system 200. Method steps are also illustrated in FIG. 1 by arrows COLLECT, ACQUIRE, STORE, PROCESS, FORWARD.

086 In the exemplary implementation, step collecting 410 is performed by service module 310; step acquiring 420 is performed by acquisition module 320; step storing 430 is performed by knowledge module 330; and steps processing 441 and forwarding 442 are executed by inference module 340. Modules 310-340 have been explained in connection with FIGS. 1-6.

087 In the exemplary implementation, steps collecting 410, acquiring 420, storing 430, processing 441 and forwarding 442 are performed for main system 200 that has a client/server configuration with database 210, application server 220, and front-end server 230. Steps collecting 410, acquiring 420, storing 430, processing 441 and forwarding 442 are performed in modules 310, 320, 330, 340 (of auxiliary system 300) that are arranged in parallel to main system 200.

088 Steps acquiring 420 knowledge representations R and forwarding 442 solutions S comprise to operate a user-interface in front-

Knowledge module 330 makes the knowledge representations R selectively available or non-available according to a selected context for subsequent step processing 441. Knowledge module 330 distinguishes context between primary context and secondary context. Knowledge module 330 stores knowledge representations R in database 210 with entries for specific problem P symptoms and corresponding solutions S. Knowledge module 330 stores knowledge representations R in database 210 with entries for predefined solutions identification rules. Knowledge module 330 stores knowledge representations R in a plurality of tables in database 210.

092 The following is optional for step processing 441: Inference module 340 performs an action such as to: identify the solutions S form a set of predefined advices of the application A, identify the solutions S by applying knowledge representations R in a sequential order, identify the solutions S by applying knowledge representations R in a hierarchical order, identify the solutions S by applying knowledge representations R in a dynamically adaptive order, communicate questions to user 1000 by composing the questions from predefined passages provided by application A, analyse responses that user 1000 enters in natural language.

093 Optionally, systems 200/300 cooperate with service system 500 (cf. FIG. 7): While executing any of steps collecting 410, acquiring 420, storing 430, processing 441 and forwarding 442, auxiliary system 300 conditionally forwards problem P data in combination with solutions S to service system 500. In the alternative, auxiliary system 300 forwards problem P data and solutions S for further analysis by a human technician. Auxiliary system 300 forwards problem P data and solutions S in a format that allows analysis by an expert system, for example

by service system 500.

094 In short, executing method 400 depends on a variety of circumstances. FIGS. 9-10 give exemplary overviews for scenarios that develop dynamically and that adapt the particular circumstances.

095 For illustration, exemplary scenarios refer to interaction (FIG. 10), context (FIG. 11) and distribution of problem collecting and solution processing (FIG. 12). The following description denotes queries by question marks.

096 FIG. 10 illustrates a simplified scenario that considers interaction, and thereby distinguishes to automatically evaluate the problem and to semi-automatically evaluate the problem. Interaction occurs among systems 200, 300 and 500 and human user 1000.

097 (10) Is interaction between main system 200 and auxiliary system 300 (or service system 500) required? The yes/no answer could depend on the performance during collecting or processing steps.

098 (30) If no, system 200 continues to automatically evaluate the problem or continues with normal operation, usually in the absence of any problems.

099 (20) If yes (interaction required): What is the interaction type: user/system (U/S) interaction (e.g., 200, 300 or 500 with 1000) or system/system (S/S) interaction (e.g., 200 with 300, 200 with 500, 300 with 500)?

0100 (21) If user/system interaction, is the interaction initiated by a user or initiated by a system?

0101 (22) If initiated by a user, for example, user 1000 detects a problem and starts a voluntary dialog with auxiliary system 300 or with system 500 at any time. A good

opportunity for starting is to press a specialized button ("PROBLEM ANALYSIS") when viewing an error message.

0102 (23) If initiated by a system, for example, system 300 collects problem related data (D) by providing a mandatory dialog (system 300 with user 1000).

0103 (24) If system/system interaction, is the interaction initiated by a user or initiated by a system?

0104 (25) If initiated by a user, for example, user 1000 starts the operation of 200/300 (cf. description method 400) or the operation of 200/500.

0105 (26) If initiated by a system, for example, system 200/500 starts its operation automatically.

0106 The query order can be modified: the query for U/S or S/S interaction (10) could follow the initializing queries. U/S and S/S interactions and initializations can be related to each other.

0107 FIG. 11 illustrates a simplified scenario that considers primary and secondary context.

0108 (10) Processing problem data D to identify context (i.e. first context) and versions, thereby using lexicon 331 (cf. FIG. 5).

0109 (20) Selecting knowledge representations R for that context (or version).

0110 (30) Processing problem data D and knowledge representations R to find solutions S.

0111 (40) Querying for the existence of a solution and finishing if solution exists.

0112 (50) Repeating processing to identify further context (i.e. second or higher context, or other versions) and querying until a solution S is identified until all

contexts have been considered.

0113 FIG. 12 illustrates a simplified scenario that considers the distribution of problem collecting and solution processing for the various systems. The scenario is useful for distributed systems with main system 200, auxiliary system 300, and service system 500 (cf. FIGS. 7-8). Depending on the availability of knowledge representations R (in auxiliary system 200 and in service system 500) that match to problem data D, the problem is automatically evaluated and solution S is identified by auxiliary system 200 or service system 500, or the problem is manually evaluated and the solution S is found by a human (e.g., user or technician). Modifications to the scenario (semi-automatically evaluating and solving) are also possible. The scenario substantially has the following phases:

0114 (10) Detecting the problem in main system 200.

0115 (20) Processing data D with R by auxiliary system 300 to identify a solution S (cf. method 400).

0116 (30) If processing successes to a solution S, solving the problem by auxiliary system 300.

0117 (40) If processing fails (no solution), forwarding data D to service system 500 (optionally enhancing D as described above).

0118 (50) Processing data D with R by service system 500 to identify a solution S (applying method 400 analogously).

0119 (60) If processing successes to a solution S, utilizing S (i.e. solve the problem) by service system 500 (or by auxiliary system 300 that is instructed by service system 500).

0120 (65) Optionally, supplying new R to auxiliary system (for finding a final solution by the auxiliary system).

0121 (70) If processing fails, evaluating the problem and finding S by a human.

0122 Once a solution S has been identified, it can be applied to actually solve the problem in a similar scenario.

0123 It is within the scope of the invention to superimpose such and other scenarios, for example, to have a scenario with interaction queries, with context consideration and with distribution.

0124 FIG. 13 illustrates simplified exemplary scenarios for solving problems in main system 200. Phases are given in top-down direction. Phases on the left side belong to the first exemplary scenario with automatically solving by the auxiliary system (illustrated on the left); phases on the right side belong to semi-automatically solving by forwarding enhanced data to service system 500. Processing phases are similar and therefore illustrated for both sides.

0125 As illustrated for the first scenario, main system 200 reports a problem (i.e. problem P); auxiliary system 300 collects the data (cf. step 410) to analyze the problem environment (i.e., operating system; versions; tables); system 300 enhances the data by the problem environment; system 300 processes data (D) and knowledge representations (R) (problem analysis) to identify the solution (S). (cf. FIG. 9, (10)(30))

0126 As illustrated for the second scenario, user 1000 initiates problem diagnosis, for example, main system 200 reports a problem (i.e. P). Auxiliary system 300 processes data (D) and knowledge representations (R), but does not identify a solution (e.g., R insufficient, D unknown). The user (or a computer specialist) manually collects further problem related data (e.g., about operating system); starts processing again to

enhance the problem data (D) by environment data; re-processing by system 300 does however not result in a solution (S). Therefore, the user decides to forward the enhanced problem data to service system 300. Service system then starts to re-evaluate the problem, usually with a more comprehensive set of knowledge representations (R) and solutions (S). (cf. FIG. 9, (10), (20), (21), (22), (10), (20), (24), (26))

0127 FIGS. 9-13 concentrate on examples. Further scenarios can be implemented, using other yes/no queries or case distinctions. Examples are explained in the following.

0128 (a) Is there a need to manually input problem data D? For some cases, especially for rare problems, preparing comprehensive knowledge representations R is not possible (or too expensive). The user can enter data via dialogs or other user interface elements.

0129 (b) Does knowledge module 330 has a sufficient number of knowledge representations R? The number is usually sufficient if processing (step 441) results in solutions S. The number is usually insufficient if processing does not result in solutions S. In that case, activating acquisition module 320 and knowledge module 330 is possible to add or modify knowledge representations R.

0130 (c) Does knowledge module 330 need to obtain further knowledge representations R? This query is related to the previous one. Representations (R) can be obtained from distributed systems. For example, system 300 can obtain R from system 500. This is convenient, for example, if system 300 operates at a customer site (occasional update) and system 500 operates at the site of a system manufacturer (daily update).

0131 (d) Are there updates available for modules 310, 320, 330, 340, for knowledge representations (R) or the like? Again, updates can be loaded from system 500 to system 300.

0132 (e) Is human support service available in a daylight time-zone to solve a problem in a night-light time-zone? If service systems 500 and specialists (i.e. service engineer 1002) are required, then system 200/300 could send a request to system 500 in a daylight time-zone. A 24-hour-service can be established with specialists working during daylight hours.

0133 (f) Is there an emergency in solving the problem or is there allowable waiting time? Are there 2 or more problems with different urgency or priority levels? Prioritizing adds value; solutions to high-profile problems could be searched in parallel by system 300 and by system 500.

0134 (g) Did the problem cause immediate symptoms? Some problems show symptoms only if they have become severe (e.g., a table with data overflow). Auxiliary system 300 can evaluate the performance of the main system to find problems before they appear to the user. Conveniently, systems 300 or 500 operate in the background to identify hidden problems and operate in the foreground (i.e. with interaction) to solve visible problems.

0135 (h) Having the solution identified, is there a predefined instructions sequence assigned to that solutions to automatically solve the problem? Automatic (or semi-automatic) problem solving according to predefined instructions could follow processing.

0136 (i) It the problem classified in a particular error class? Problems and their corresponding solutions can be classified in great variety. Exemplary classes are:

EPO - Munich
55
31. Okt. 2002

Claims

1. Distributed computer system (201/202/500) comprising: a first main system (201) and a second main system (202), both to execute applications (A) in cooperation with a human user (1000), a service system (500) to evaluate problems (P) in the main systems (201, 202), the service system (500) having a service module (510) to collect problem related data (D) from the main systems (201, 202), an acquisition module (520) to acquire knowledge representations (R), a knowledge module (530) to store the knowledge representations (R), an inference module (540) for processing problem related data (D) with knowledge representations (R) to identify solutions (S), the inference module (540) forwarding the solutions (S) through the service module (510) to the main systems (201, 201).

2. The computer system (201/202/500) of claim 1, wherein the first and second main systems (201, 202) have first and second auxiliary systems with auxiliary knowledge representations to evaluate problems (P) in the main systems (201, 202) and to escalate problem evaluation to the service system (500).

3. The computer system (201/202/500) of claim 2, wherein the knowledge representations in the service system (500) are enhanced in comparison to the auxiliary knowledge representations in the first and second auxiliary systems.

4. The computer system (201/202/500) of claim 3, wherein the knowledge representations are enhanced in volume, actuality and complexity.

11. Inference module (x40) with expertise functionality for evaluating problems (P) in first and second main computer systems (201, 202) that execute an application (A), wherein the inference module (x40) is adapted to process problem related data D with knowledge representations R to identify solutions S, the inference module (x40) characterized in that the inference module (x40) is part of a service system (500) that receives problem related data (D) from the first and second main systems (201, 202) of different versions over a network, wherein the inference module (x40) applies the knowledge representations (R) for both main systems (201, 202) and distinguishes version differences of the main systems (201, 202) by looking up in a check lexicon (331).

EPO - Munich
55
31. Okt. 2002

Abstract of the Disclosure

A distributed computer system (201/202/500) has a first main system (201) and a second main system (202) that execute applications (A) in cooperation with human users. A service system (500) is an expert system to evaluate problems in the main systems (201, 202). The main systems (201, 202) have auxiliary systems with to evaluate problems in the main systems (201, 202) and to escalate problem evaluation to the service system (500). The service system (500) provides expertise for customers 1 and 2 that operate the main system (201, 202).

EPO - Munich
55
31. Okt. 2002

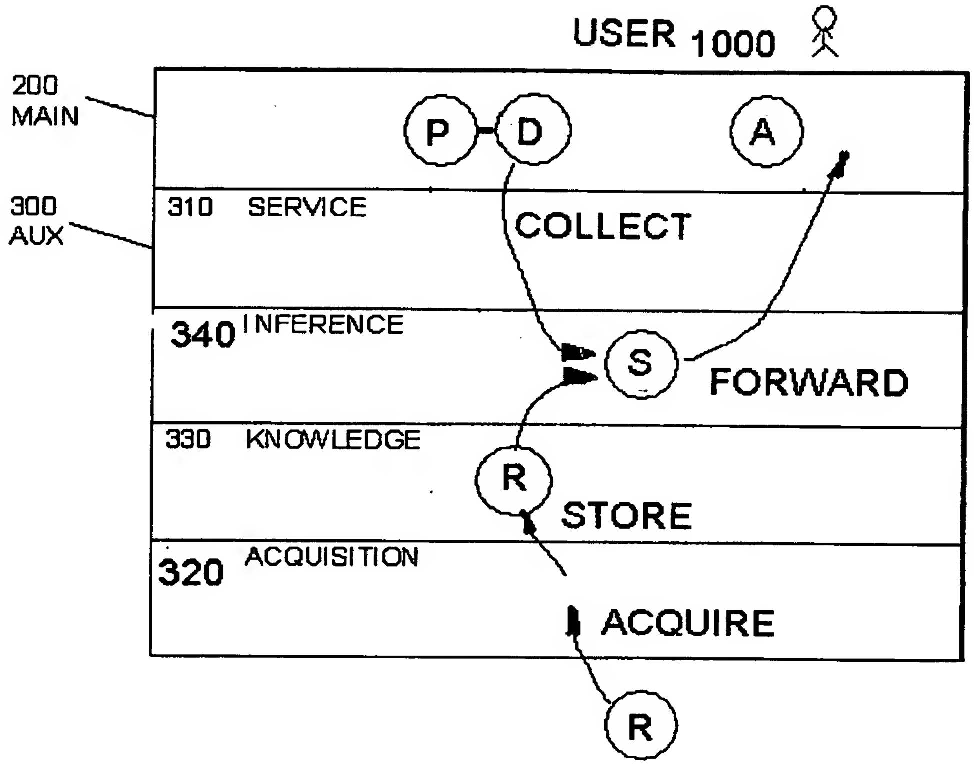
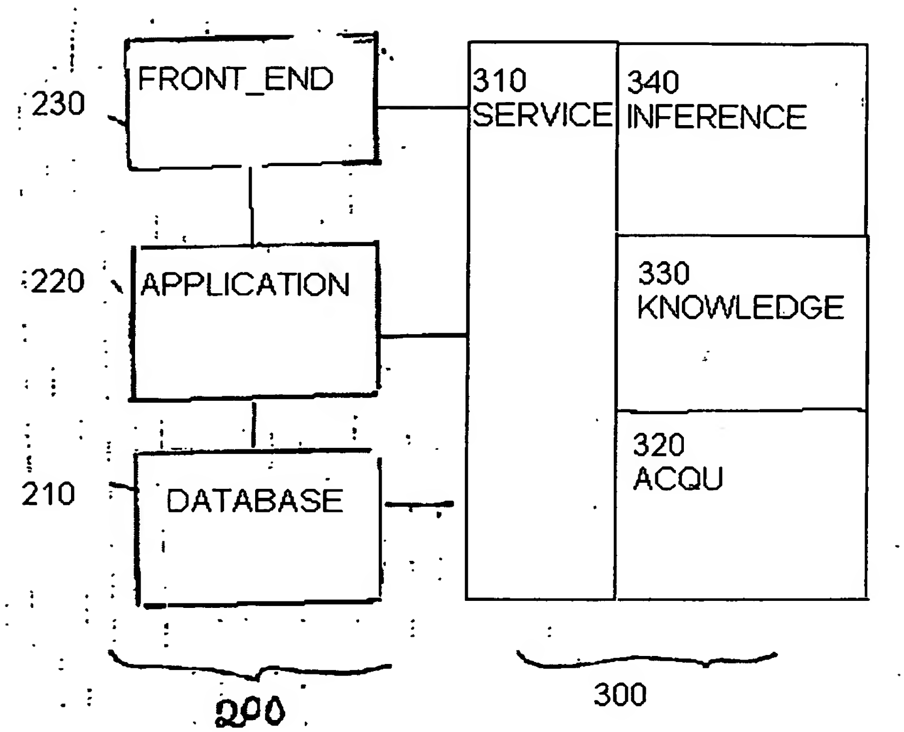
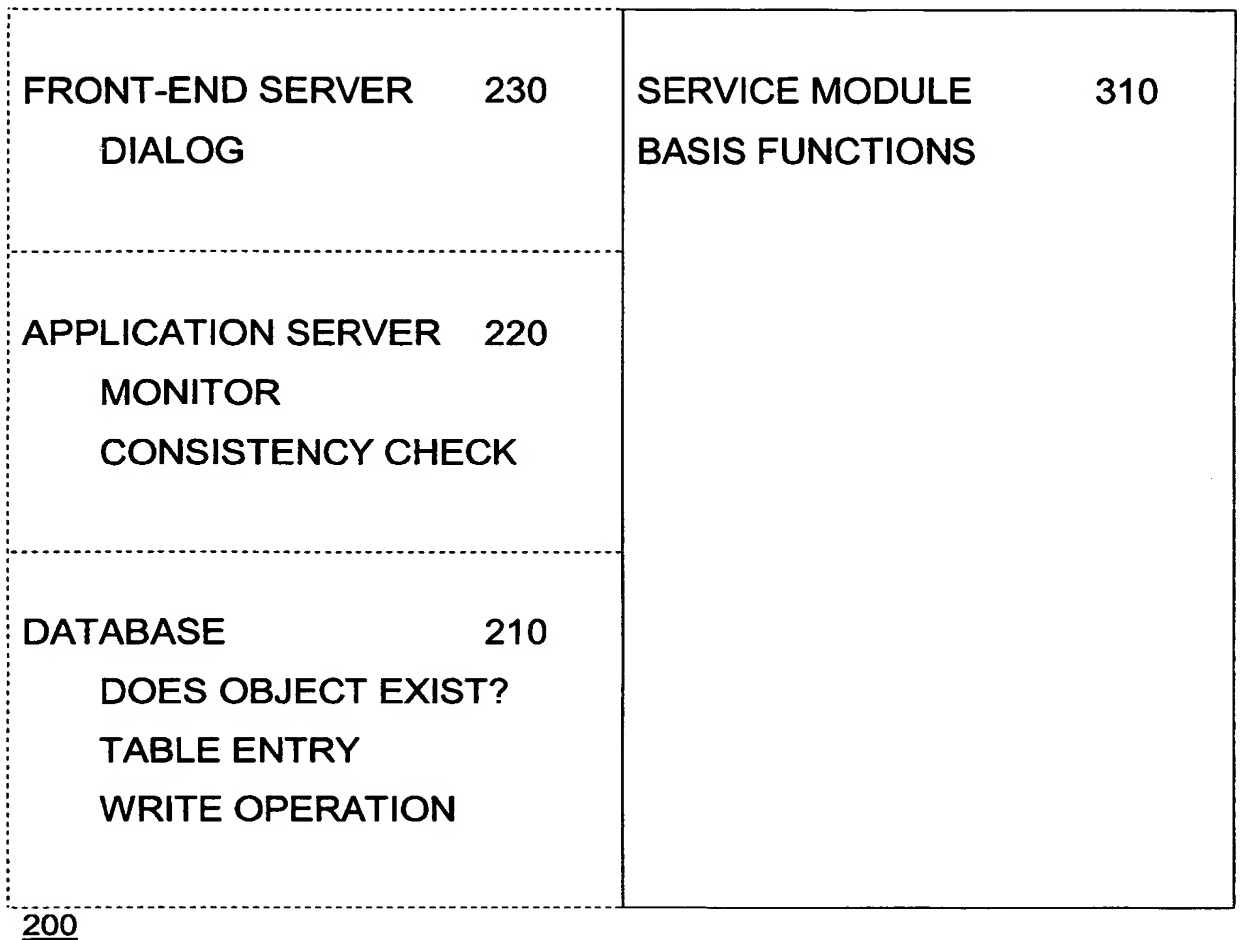


FIG. 1



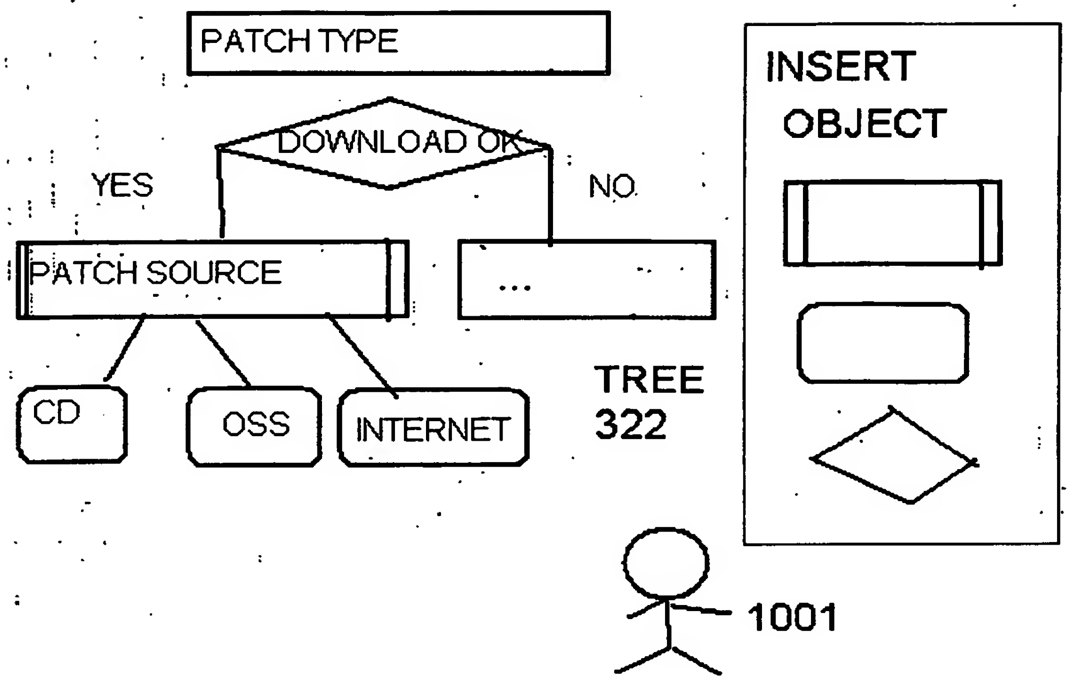
COMPUTER SYSTEM 200/300

FIG. 2



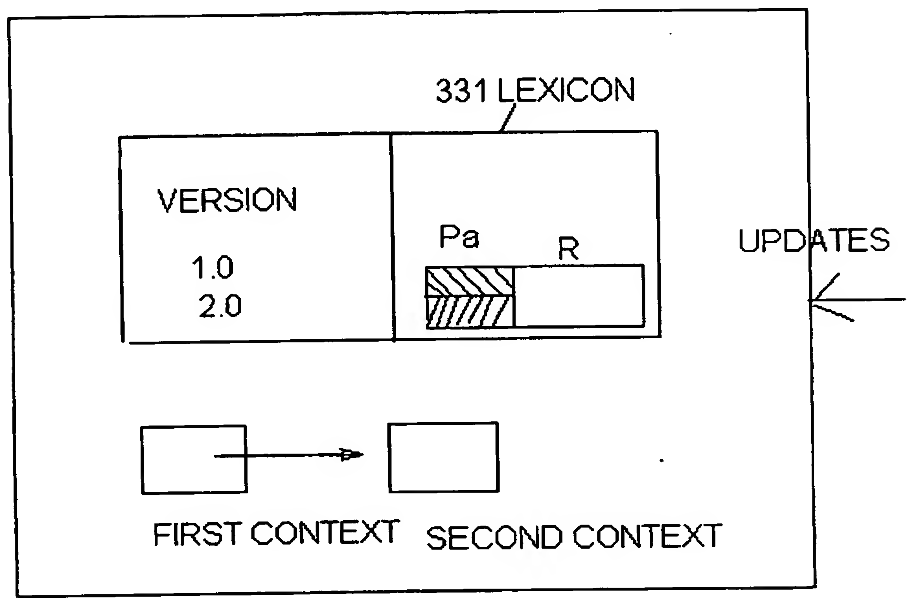
SERVICE MODULE 310

FIG. 3



ACQUISITION MODULE 320

FIG. 4



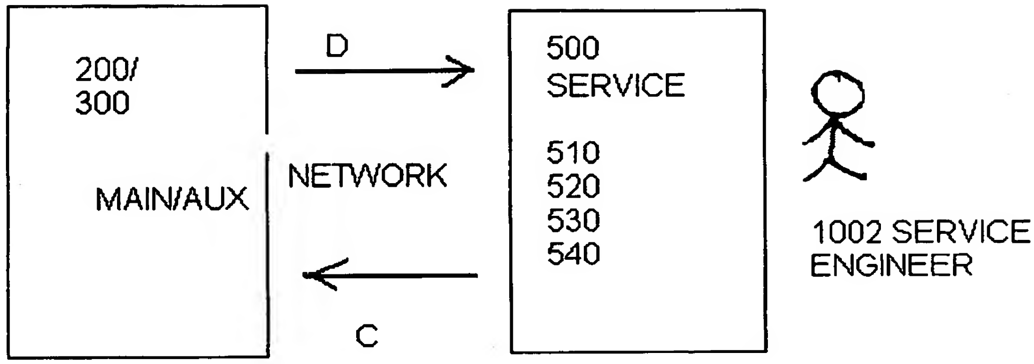
KNOWLEDGE MODULE 330

FIG. 5

SOLUTION IDENTIFIER
SET OF ADVICES
R TO APPLY
R1, R2, R3
R1, R21, R21, R31, R32
IF THEN
Q1, Q2, Q3 RESERVOIR
LANGUAGE ANALYZER

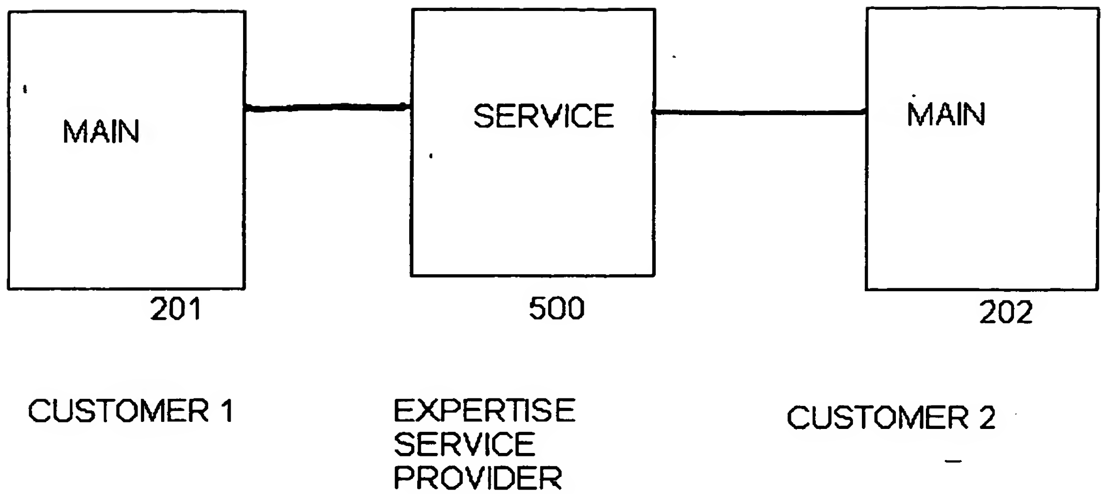
INFERENCE MODULE 340

FIG. 6



FIRST DISTRIBUTED SYSTEM LANDSCAPE

FIG. 7



SECOND DISTRIBUTED SYSTEM LANDSCAPE

FIG. 8

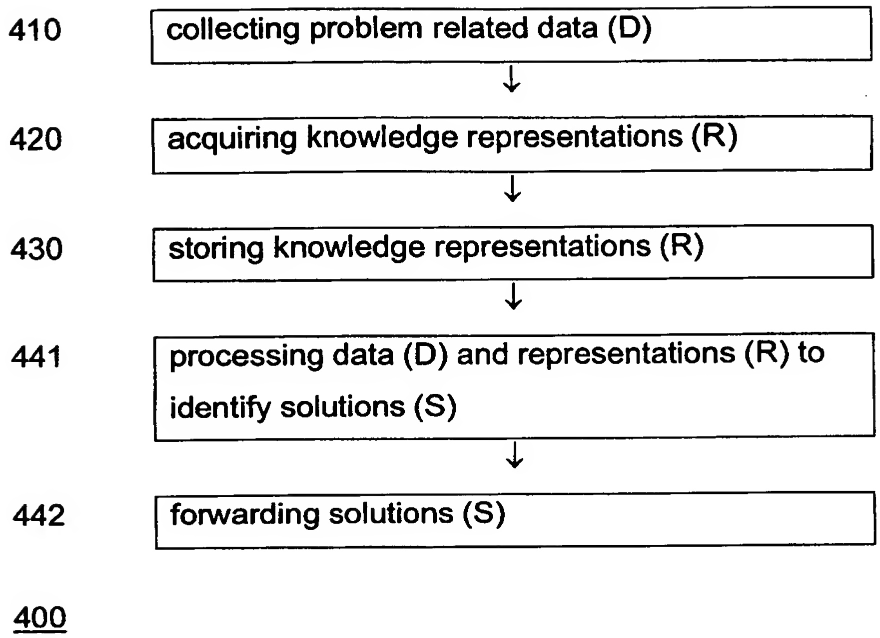

410
collecting problem related data (D)
420
acquiring knowledge representations (R)
430
storing knowledge representations (R)
441
processing data (D) and representations (R) to identify solutions (S)
442
forwarding solutions (S)
400

FIG. 9

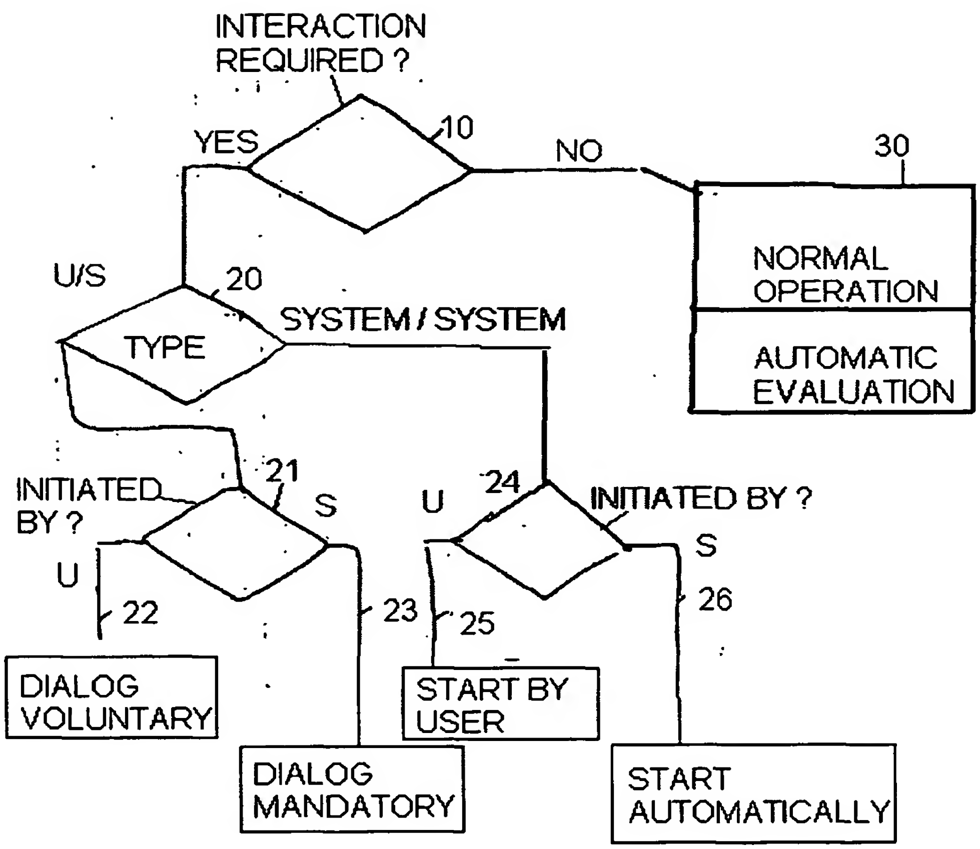

INTERACTION REQUIRED ?
10
YES
NO
30
NORMAL OPERATION
AUTOMATIC EVALUATION
U/S
20
TYPE
SYSTEM / SYSTEM
21
INITIATED BY ?
S
U
24
INITIATED BY ?
U
S
22
23
25
26
DIALOG VOLUNTARY
DIALOG MANDATORY
START BY USER
START AUTOMATICALLY

FIG. 10

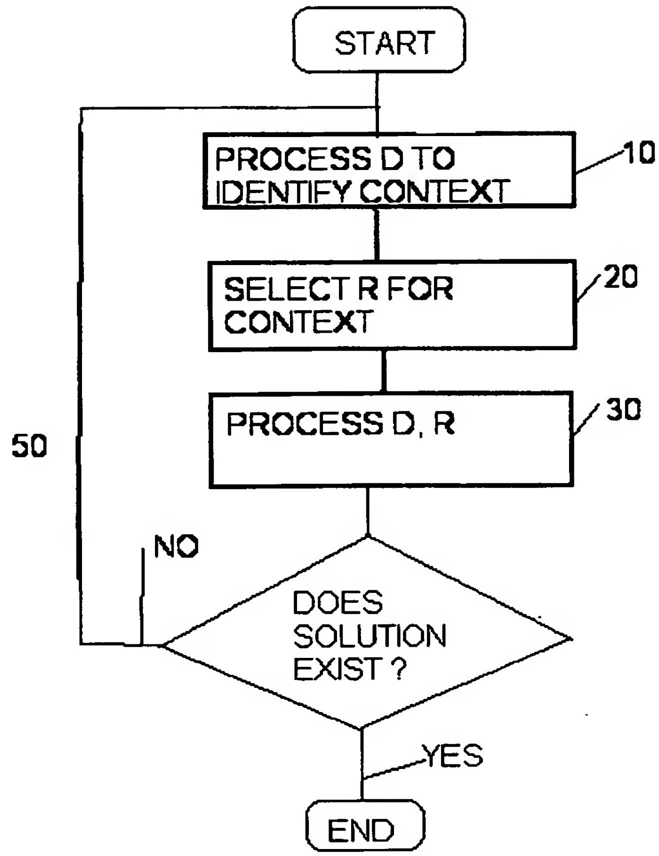

START
PROCESS D TO IDENTIFY CONTEXT
10
SELECT R FOR CONTEXT
20
PROCESS D, R
30
50
NO
DOES SOLUTION EXIST ?
YES
END

FIG. 11

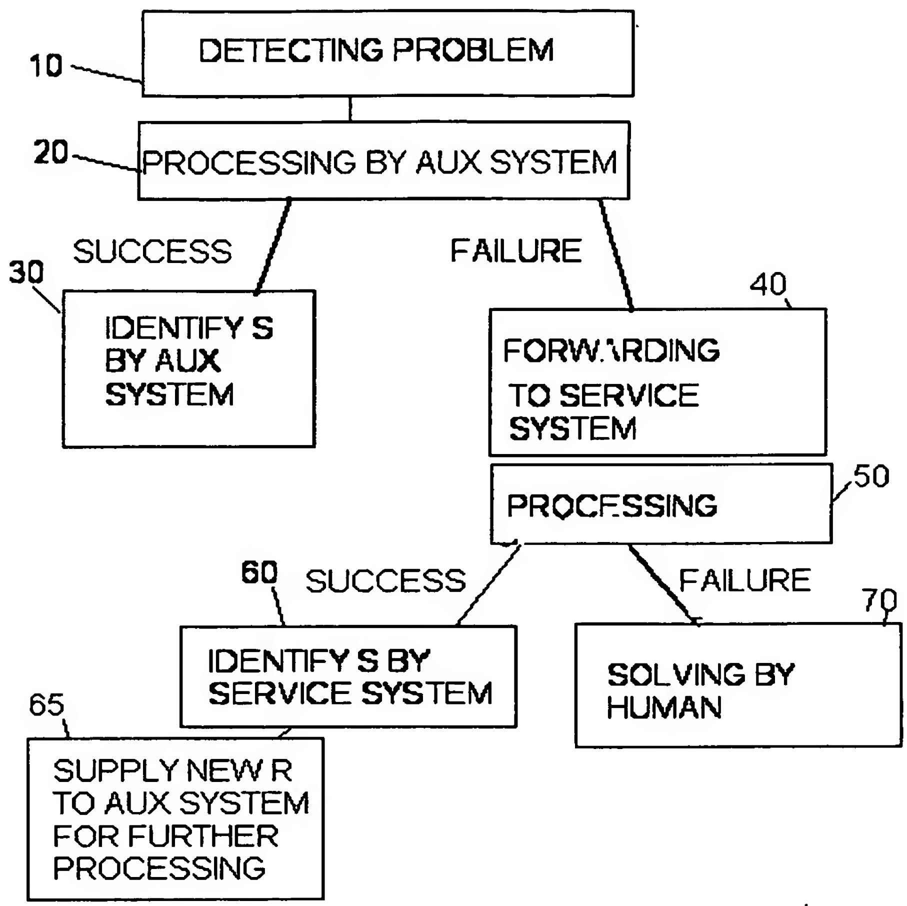

10
DETECTING PROBLEM
20
PROCESSING BY AUX SYSTEM
SUCCESS
FAILURE
30
IDENTIFY S BY AUX SYSTEM
40
FORWARDING TO SERVICE SYSTEM
PROCESSING
50
60
SUCCESS
FAILURE
70
IDENTIFY S BY SERVICE SYSTEM
SOLVING BY HUMAN
65
SUPPLY NEW R TO AUX SYSTEM FOR FURTHER PROCESSING

FIG. 12

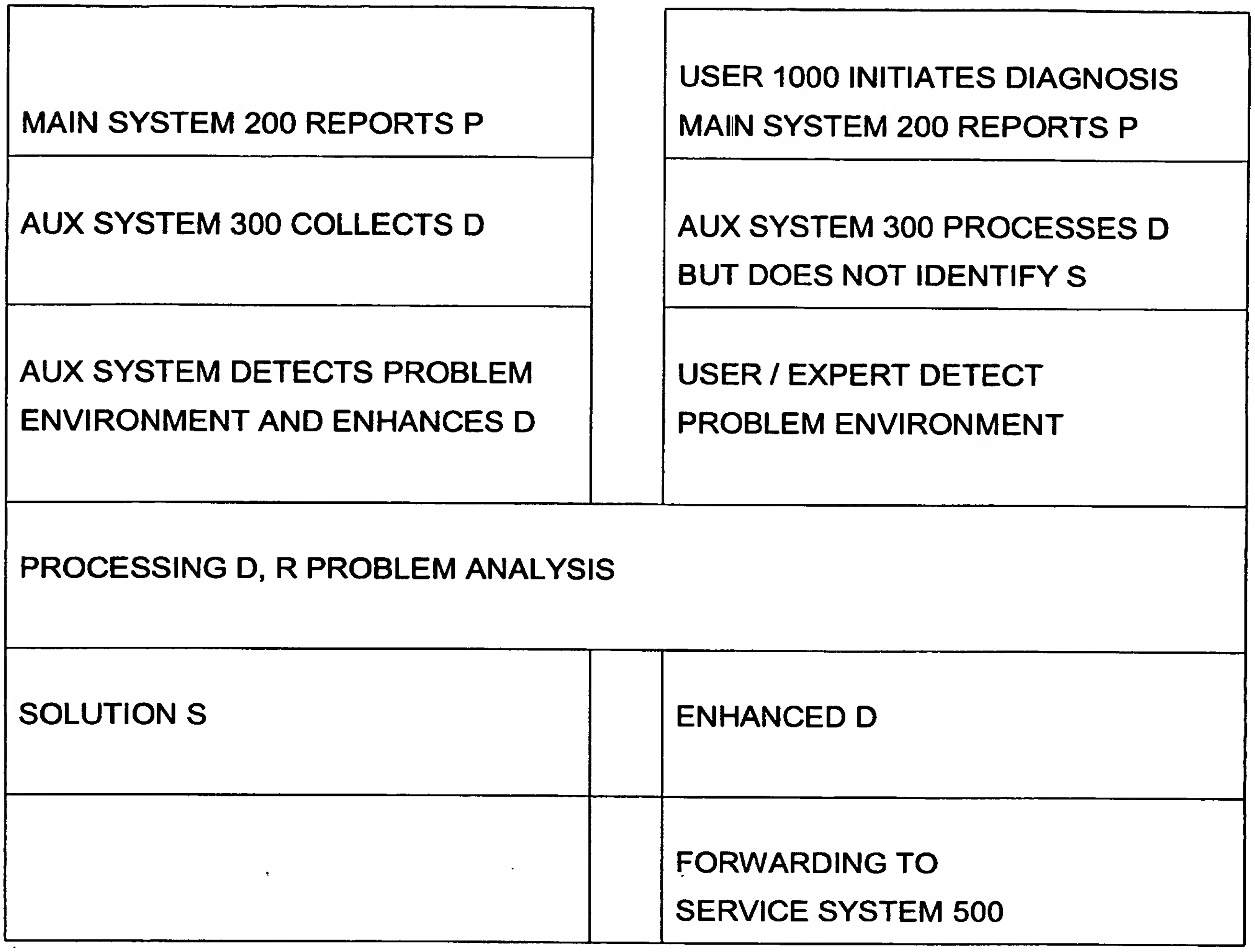

MAIN SYSTEM 200 REPORTS P
USER 1000 INITIATES DIAGNOSIS
MAIN SYSTEM 200 REPORTS P
AUX SYSTEM 300 COLLECTS D
AUX SYSTEM 300 PROCESSES D
BUT DOES NOT IDENTIFY S
AUX SYSTEM DETECTS PROBLEM
ENVIRONMENT AND ENHANCES D
USER / EXPERT DETECT
PROBLEM ENVIRONMENT
PROCESSING D, R PROBLEM ANALYSIS
SOLUTION S
ENHANCED D
FORWARDING TO
SERVICE SYSTEM 500

FIG. 13

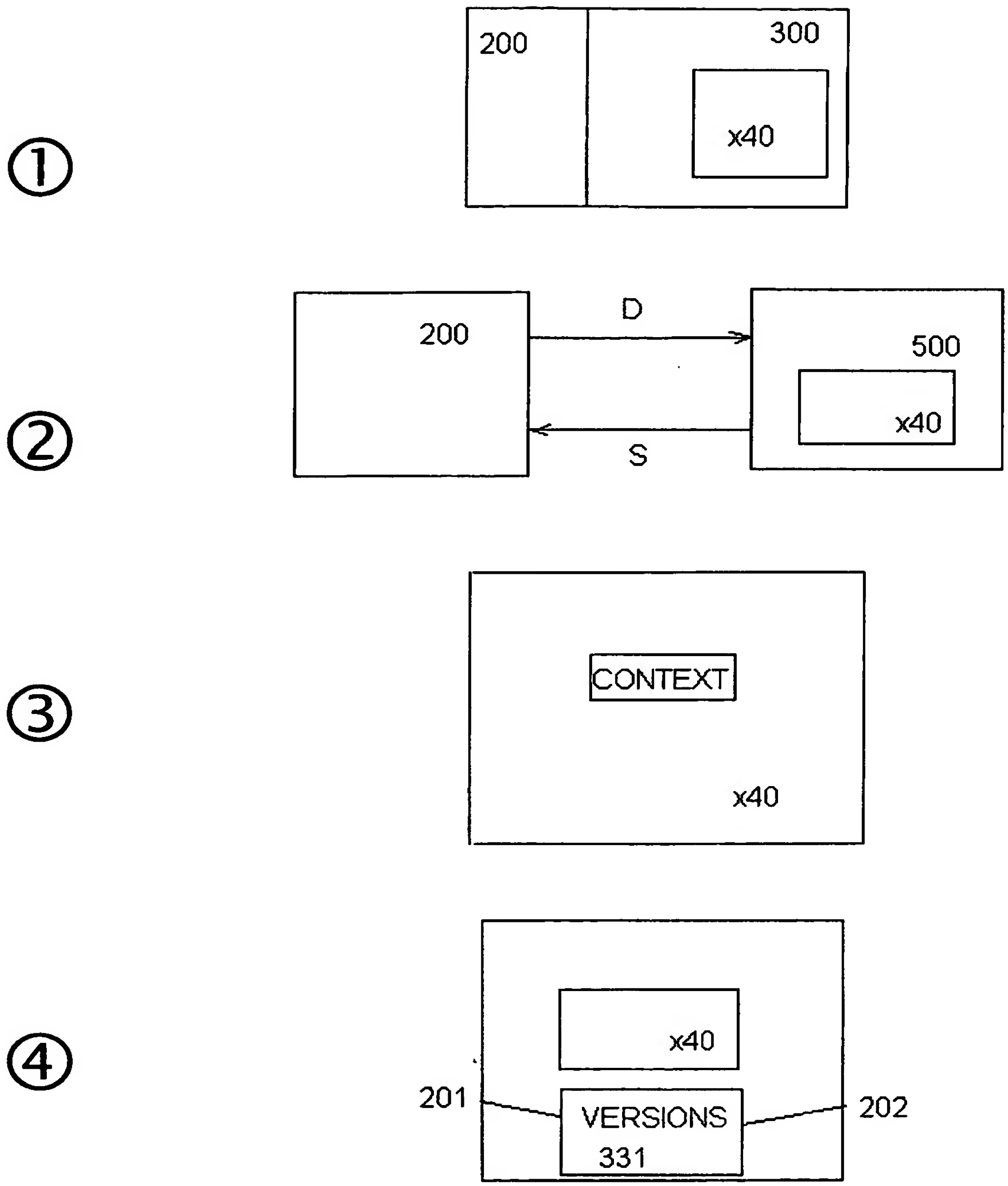

①
200
300
x40
②
200
D
S
500
x40
③
CONTEXT
x40
④
x40
201
VERSIONS
331
202

FIG. 14

EPO - Munich
55
31. Okt. 2002

Claims

1. Distributed computer system (201/202/500) comprising:
   a first main system (201) and a second main system (202), both to execute applications (A) in cooperation with a human user (1000),
   a service system (500) to evaluate problems (P) in the main systems (201, 202), the service system (500) having a service module (510) to collect problem related data (D) from the main systems (201, 202), an acquisition module (520) to acquire knowledge representations (R), a knowledge module (530) to store the knowledge representations (R), an inference module (540) for processing problem related data (D) with knowledge representations (R) to identify solutions (S), the inference module (540) forwarding the solutions (S) through the service module (510) to the main systems (201, 201).

2. The computer system (201/202/500) of claim 1, wherein the first and second main systems (201, 202) have first and second auxiliary systems with auxiliary knowledge representations to evaluate problems (P) in the main systems (201, 202) and to escalate problem evaluation to the service system (500).

3. The computer system (201/202/500) of claim 2, wherein the knowledge representations in the service system (500) are enhanced in comparison to the auxiliary knowledge representations in the first and second auxiliary systems.

4. The computer system (201/202/500) of claim 3, wherein the knowledge representations are enhanced in volume, actuality and complexity.

5. The computer system (201/202/500) of claim 2, wherein the first and second auxiliary systems forward problem data to the service system (500) after preliminary data analysis by processing with the auxiliary knowledge representations.

6. The computer system (201/202/500) of claim 2, wherein the services system (500) updates the auxiliary knowledge representations in first and second auxiliary systems.

7. The computer system (201/202/500) of claim 2, wherein the first and second service systems each have a service module (510) to collect problem related data (D) from the main systems (201, 202), an acquisition module (520) to acquire knowledge representations (R), a knowledge module (530) to store the knowledge representations (R), an inference module (540) for processing problem related data (D) with knowledge representations (R) to identify solutions (S), the inference module (540) for selectively forwarding the solutions (S) through the service module (510) to the main systems (201, 201) and forwarding data (D) to the service system (500).

8. The computer system (201/202/500) of claim 1, wherein the inference module (x40) applies the knowledge representations (R) for both main systems (201, 202) and distinguishes version differences of the main systems by looking up in a check lexicon (331).

9. Method for solving problem in at least one main computer system by expert systems, comprising:
detecting the problem in the main system;
processing problem related data with a first set of knowledge representations of a first expert system to search for a solution;
depending on processing results, selectively solving the problem by the first expert system or forwarding the problem related data together with search results to a second expert system with a second set of knowledge representations;
processing the problem related data, the search results and the second set of knowledge representations by the second expert system to search for the solution;
depending on processing results, selectively solving the problem by the second expert system or presenting search results of both searches and problem related data to a human.

10. Computer program product (100) comprising program code means for performing all the steps of anyone of the claims 1-9 when the program product is run on a computer.

11. Inference module (x40) with expertise functionality for evaluating problems (P) in first and second main computer systems (201, 202) that execute an application (A), wherein the inference module (x40) is adapted to process problem related data D with knowledge representations R to identify solutions S, the inference module (x40) characterized in that the inference module (x40) is part of a service system (500) that receives problem related data (D) from the first and second main systems (201, 202) of different versions over a network, wherein the inference module (x40) applies the knowledge representations (R) for both main systems (201, 202) and distinguishes version differences of the main systems (201, 202) by looking up in a check lexicon (331).

EPO - Munich
55
31. Okt. 2002

Abstract of the Disclosure

A distributed computer system (201/202/500) has a first main system (201) and a second main system (202) that execute applications (A) in cooperation with human users. A service system (500) is an expert system to evaluate problems in the main systems (201, 202). The main systems (201, 202) have auxiliary systems with to evaluate problems in the main systems (201, 202) and to escalate problem evaluation to the service system (500). The service system (500) provides expertise for customers 1 and 2 that operate the main system (201, 202).

EPO - Munich
55
31. Okt. 2002

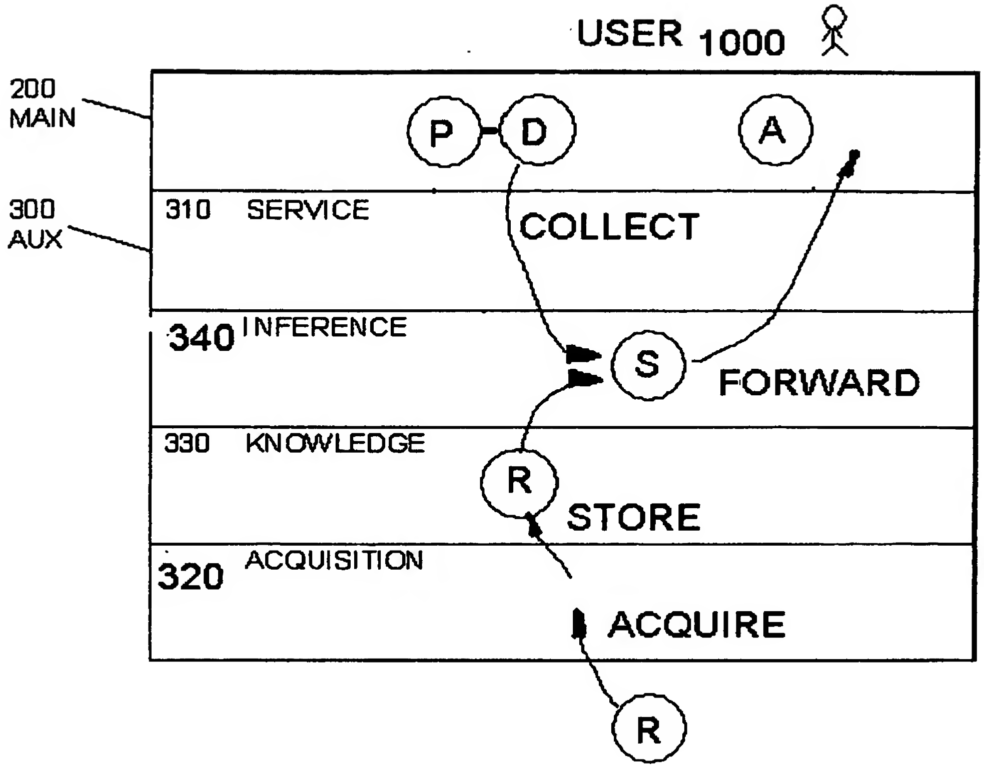
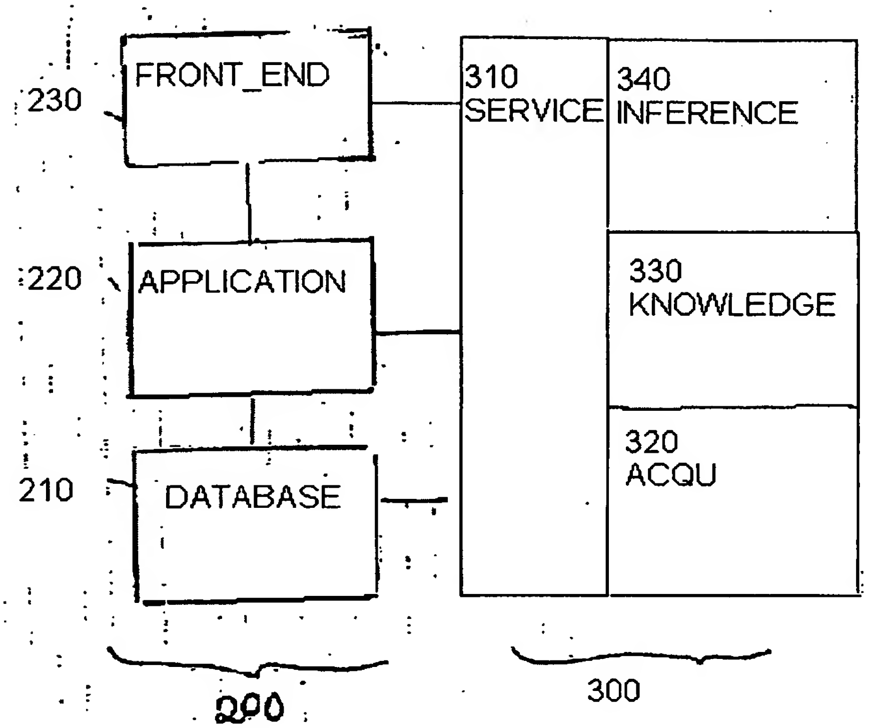
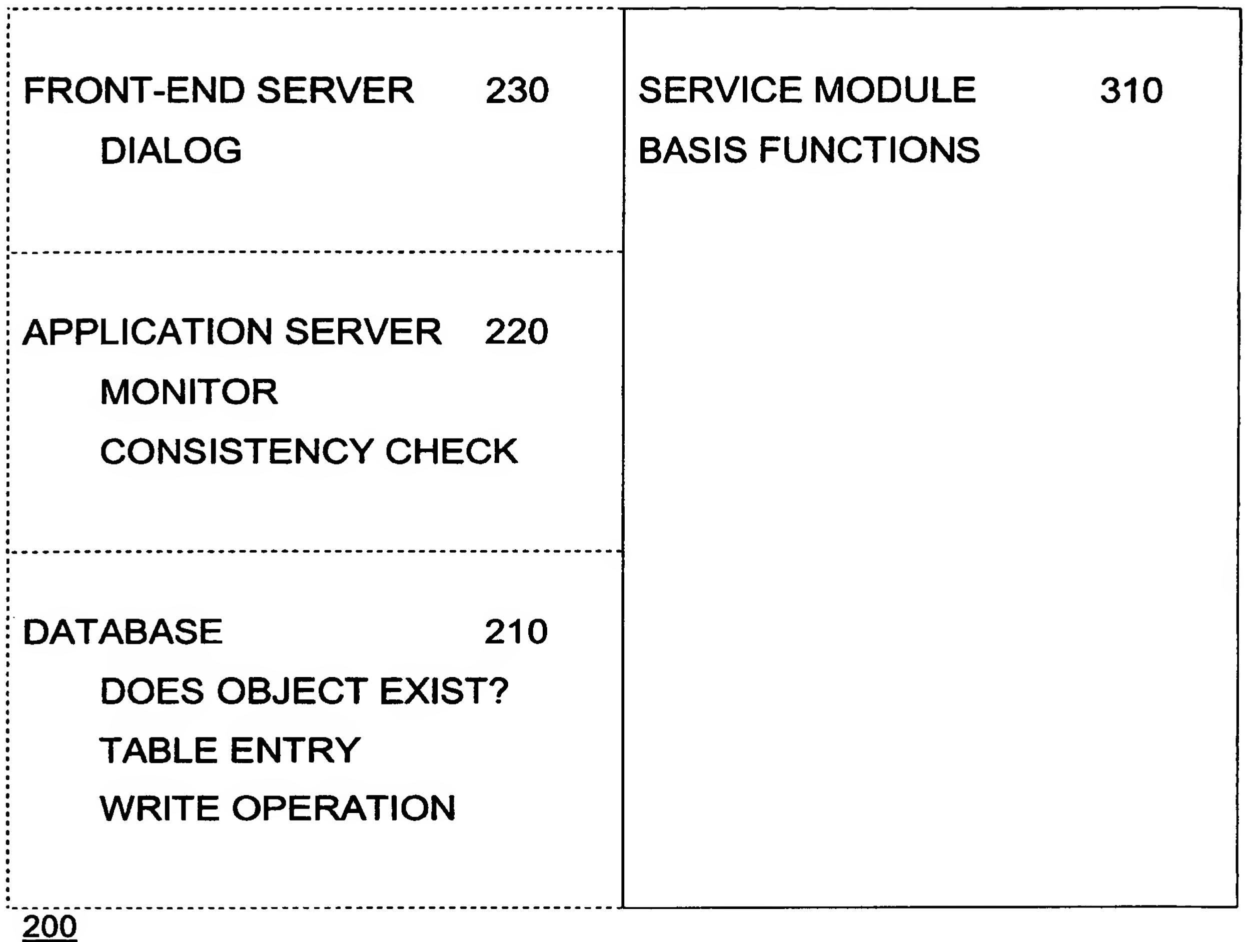


FIG. 1



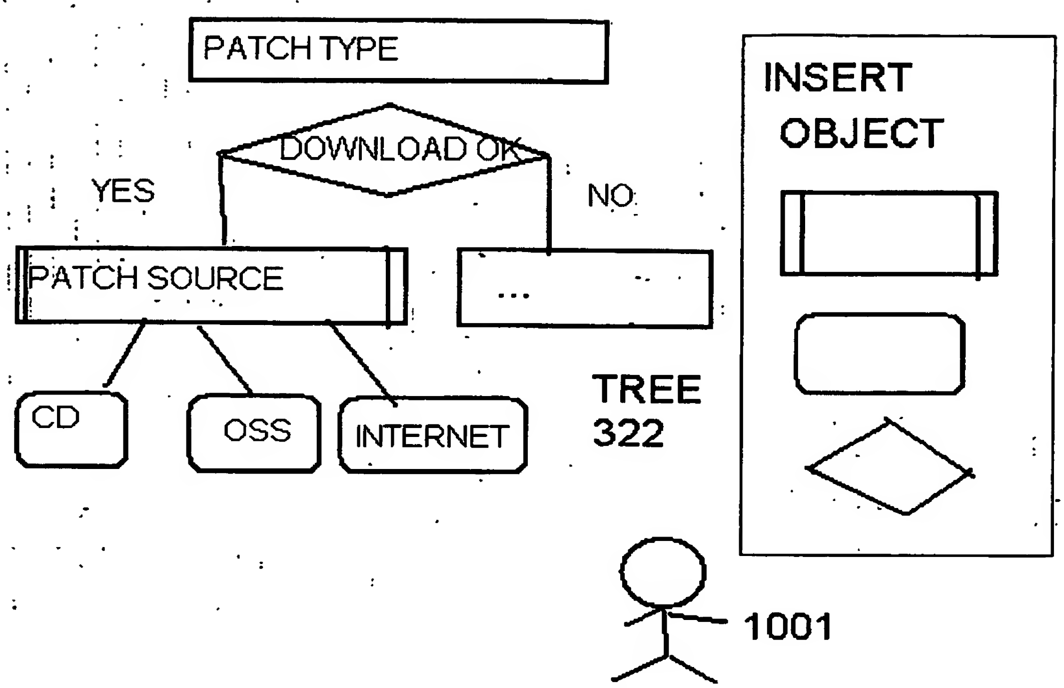
COMPUTER SYSTEM 200/300

FIG. 2



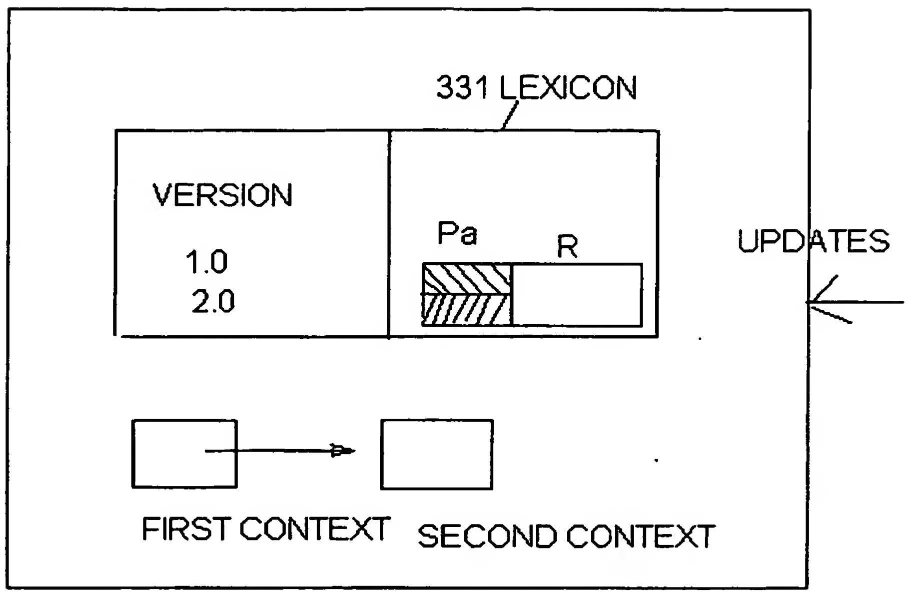
SERVICE MODULE 310

FIG. 3



ACQUISITION MODULE 320

FIG. 4



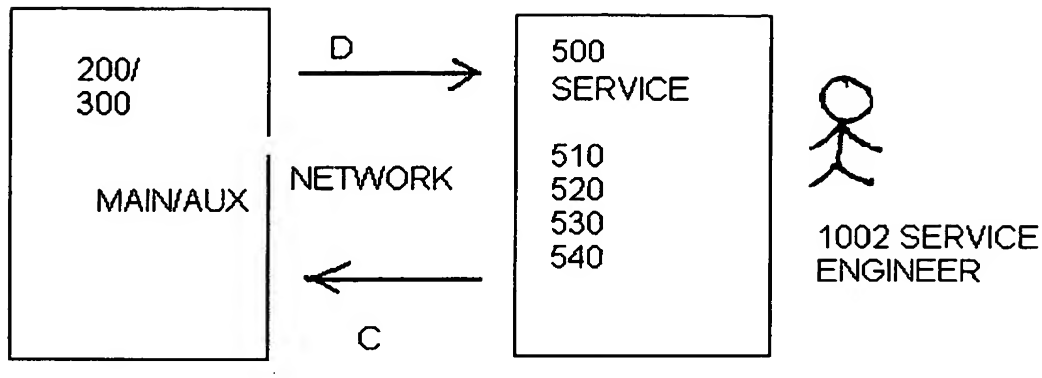
KNOWLEDGE MODULE 330

FIG. 5

SOLUTION IDENTIFIER
SET OF ADVICES
R TO APPLY
R1, R2, R3
R1, R21, R21, R31, R32
IF THEN
Q1, Q2, Q3 RESERVOIR
LANGUAGE ANALYZER

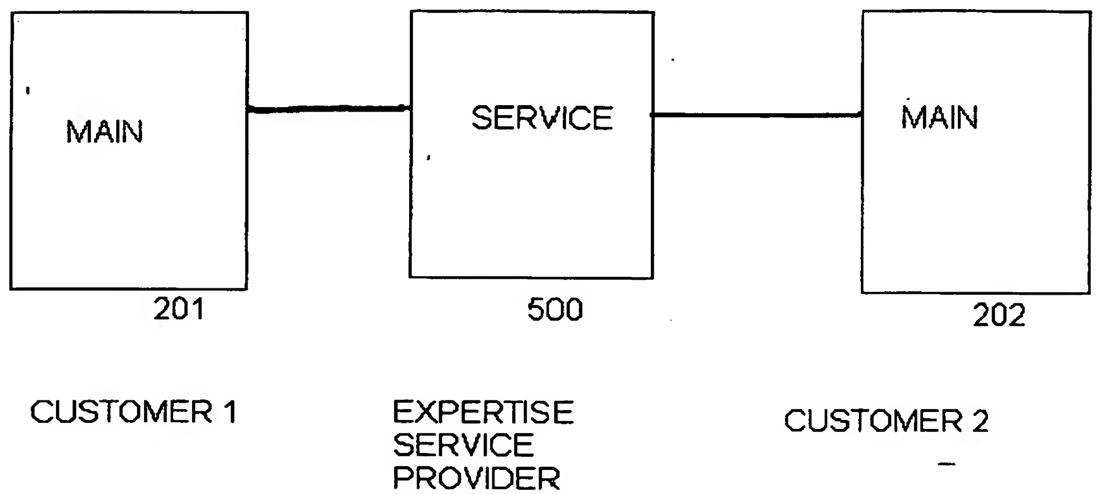
INFERENCE MODULE 340

FIG. 6



FIRST DISTRIBUTED SYSTEM LANDSCAPE

FIG. 7



SECOND DISTRIBUTED SYSTEM LANDSCAPE

FIG. 8

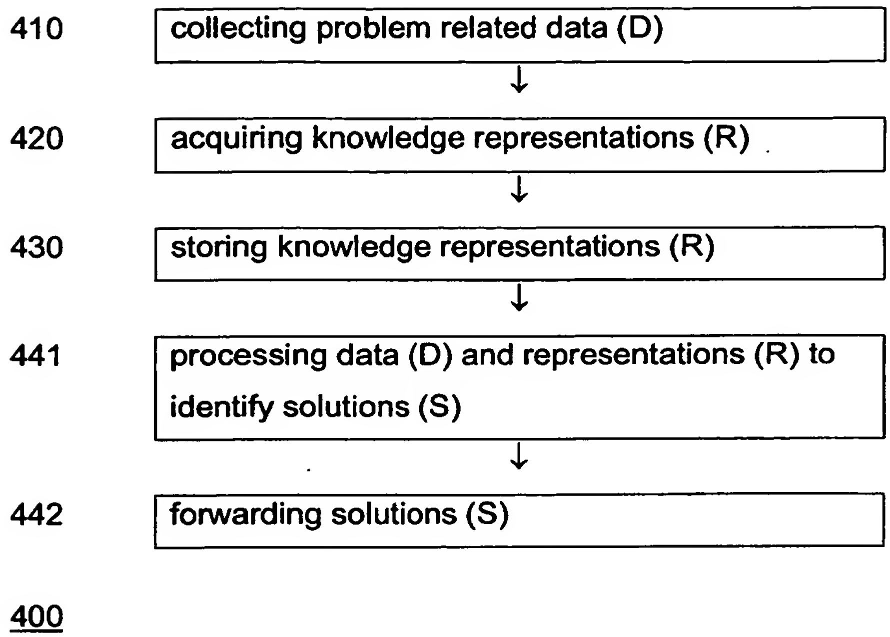

410
collecting problem related data (D)
420
acquiring knowledge representations (R)
430
storing knowledge representations (R)
441
processing data (D) and representations (R) to identify solutions (S)
442
forwarding solutions (S)
400

FIG. 9

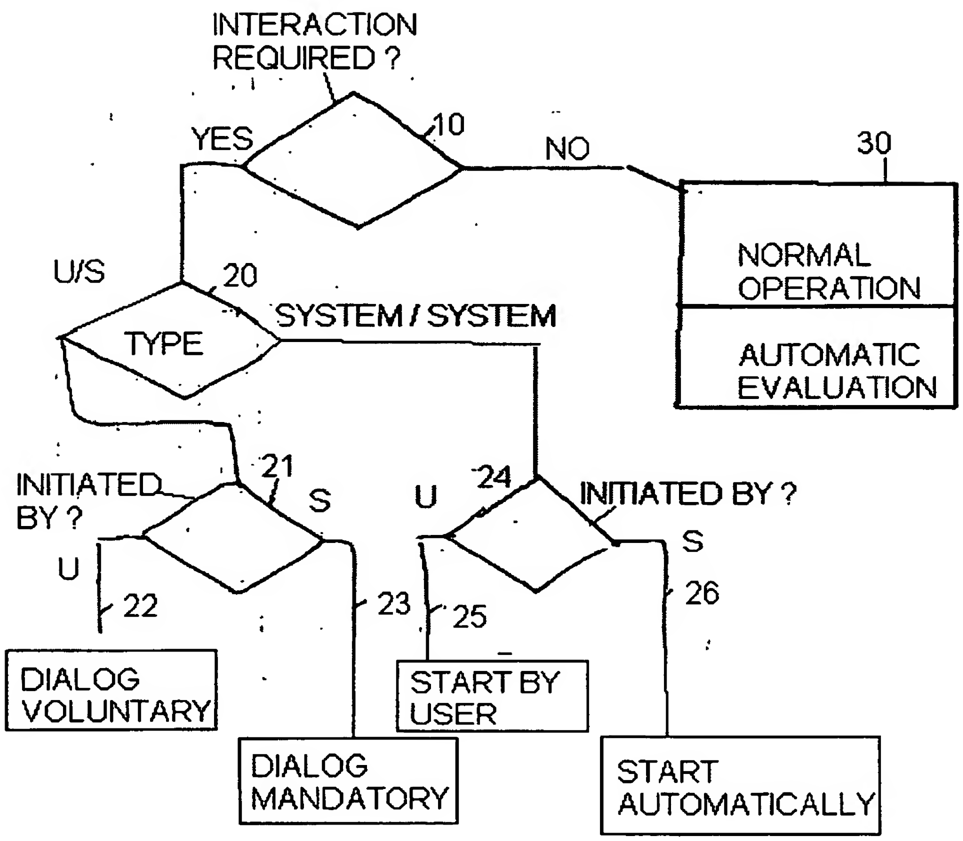

INTERACTION
REQUIRED ?
YES
10
NO
30
NORMAL
OPERATION
AUTOMATIC
EVALUATION
U/S
20
TYPE
SYSTEM / SYSTEM
INITIATED
BY ?
21
S
U
22
23
DIALOG
VOLUNTARY
DIALOG
MANDATORY
U
24
INITIATED BY ?
S
25
26
START BY
USER
START
AUTOMATICALLY

FIG. 10

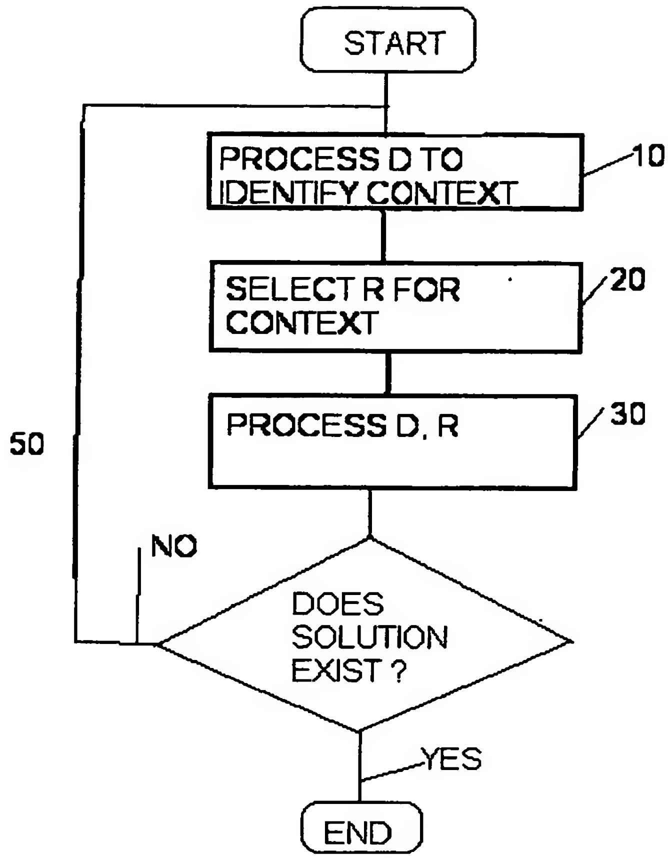

START
PROCESS D TO IDENTIFY CONTEXT
10
SELECT R FOR CONTEXT
20
PROCESS D, R
30
50
NO
DOES SOLUTION EXIST ?
YES
END

FIG. 11

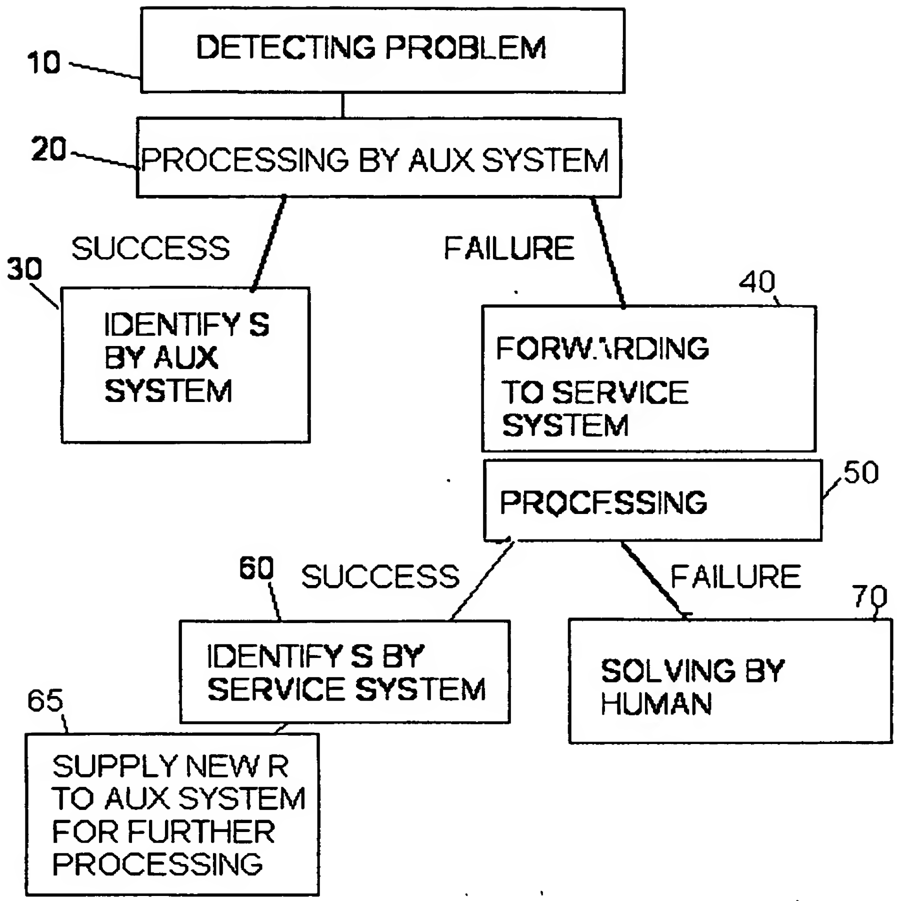
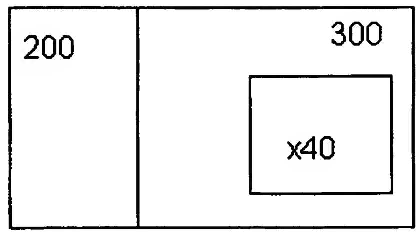

DETECTING PROBLEM
10
20
PROCESSING BY AUX SYSTEM
SUCCESS
FAILURE
30
IDENTIFY S BY AUX SYSTEM
40
FORWARDING TO SERVICE SYSTEM
PROCESSING
50
60
SUCCESS
FAILURE
70
IDENTIFY S BY SERVICE SYSTEM
SOLVING BY HUMAN
65
SUPPLY NEW R TO AUX SYSTEM FOR FURTHER PROCESSING

FIG. 12

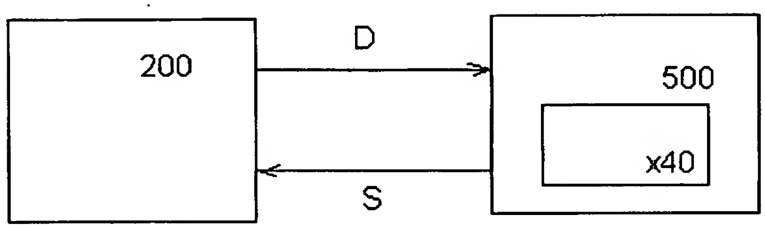
MAIN SYSTEM 200 REPORTS P	USER 1000 INITIATES DIAGNOSIS MAIN SYSTEM 200 REPORTS P
AUX SYSTEM 300 COLLECTS D	AUX SYSTEM 300 PROCESSES D BUT DOES NOT IDENTIFY S
AUX SYSTEM DETECTS PROBLEM ENVIRONMENT AND ENHANCES D	USER / EXPERT DETECT PROBLEM ENVIRONMENT
PROCESSING D, R PROBLEM ANALYSIS	
SOLUTION S	ENHANCED D
	FORWARDING TO SERVICE SYSTEM 500

FIG. 13

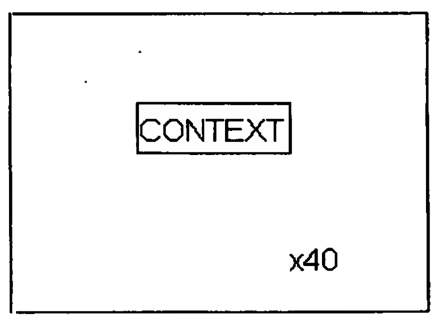
①


200
300
x40

②


200
D
S
500
x40

③


CONTEXT
x40

④

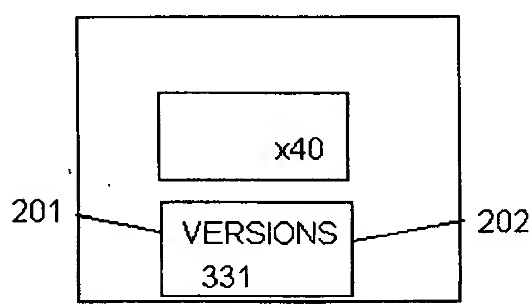

x40
201
VERSIONS
331
202

FIG. 14

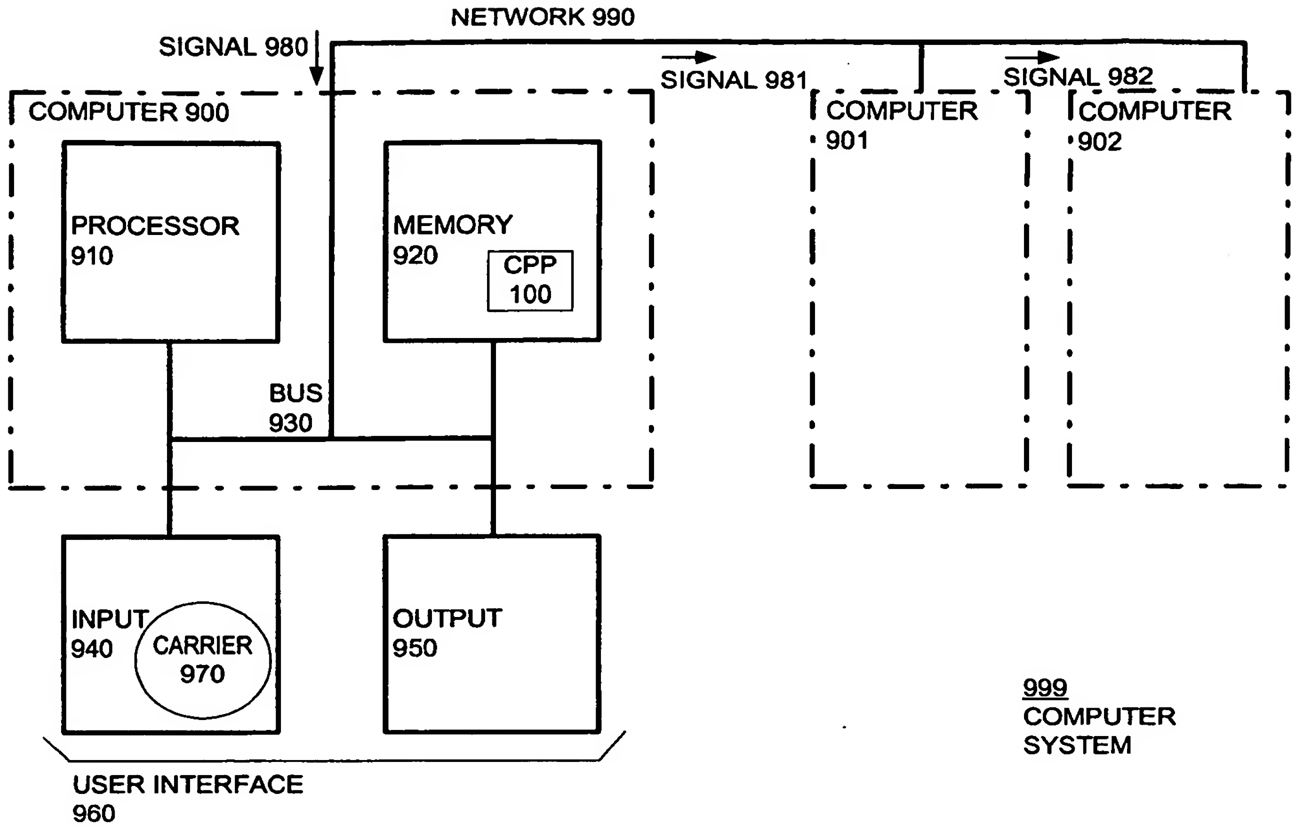

NETWORK 990
SIGNAL 980
SIGNAL 981
SIGNAL 982
COMPUTER 900
PROCESSOR 910
MEMORY 920
CPP 100
BUS 930
COMPUTER 901
COMPUTER 902
INPUT 940
CARRIER 970
OUTPUT 950
USER INTERFACE 960
999 COMPUTER SYSTEM

FIG. 15